Declaration and Initialization Solutions

Copyright © James Raynard 2021

Universal Initialization

- What is meant by universal initialization in C++?
  - Universal or brace initialization means that we put the initial value(s) of a variable in braces {}
  - This works with all built-in types and classes, provided the constructor accepts the given number of arguments
- Write a program which demonstrates universal initialization of built-in types and objects with single and multiple initial values

Copyright © James Raynard 2021

Advantages of Universal Initialization

- What are the advantages of universal initialization?
  - The same syntax can be used to initialize all types
  - Easy initialization of containers with multiple different values
  - Narrowing conversions are caught by the compiler
  - Avoids the "most vexing parse"

Copyright © James Raynard 2021

Nested template declaration

- Is this code legal in Modern C++?

```
vector<vector<int>> vec_of_vec;
```

  - Yes, it is no longer necessary to put a space between the closing brackets

Copyright © James Raynard 2021

Type alias

- What is meant by a "type alias"? Why is it useful?
  - A type alias gives another name for a type
  - It can be used to simplify code which uses complex types such as function pointers or nested containers
- Rewrite the following type alias using the Modern C++ alternative

```
typedef vector<int> IntVec;
vector<IntVec> vec_of_vec;
```

- Compile your answer to check it is correct

Copyright © James Raynard 2021

nullptr

- Modern C++ gives us nullptr as an alternative to NULL
- Are there any advantages to using nullptr?
  - NULL is a macro whose type is implementation defined
  - nullptr has a specific type which is unique to it. This avoids compiler-dependent behaviour. It could also be useful in template metaprogramming

Copyright © James Raynard 2021